Lab 9: Digital Filters in LabVIEW and Matlab

INTRODUCTION:

In Lab 8, a hardware bandpass filter was designed to remove noise from the recorded ECG signals. Sometimes software tools are employed to implement the desired filters. Compared to hardware filters, software filters can be more convenient and flexible because the user just needs to define filter parameters and the software will automatically generate the necessary filter coefficients. In this lab, software digital filters will be design in LabVIEW and Matlab/Simulink and compared.

REQUIRED PARTS AND MATERIALS:

Materials Needed

1) Oscilloscope
2) Function generator
3) DC power supply
4) Elvis DAQ interface board
5) LabVIEW software
6) Matlab software

PRELAB:

1. Print the Prelab and Lab9 Grading Sheets. Answer all of the questions in the Prelab Grading Sheet and bring the Lab9 Grading Sheet with you when you come to lab. ***The Prelab Grading Sheet must be turned in to the TA before beginning your lab assignment.***
2. Read the LABORATORY PROCEDURE before coming to lab. Note: you are not required to print the lab procedure; you can view it on the PC at your lab bench.

BACKGROUND:

Digital filters are a very important part of digital signal processing. In fact, their extraordinary performance is one of the key reasons that DSP has become so popular. Digital filters have two primary functions: signal separation and signal restoration. Signal separation is needed when a signal has been contaminated with interference, noise, or other undesirable signals. For example, when measuring the electrical activity of an unborn baby in the womb, the raw signal will likely be corrupted by the breathing and heartbeat of the mother. A filter could be used to separate these signals so that they can be individually analyzed. Signal restoration is used when a signal has been distorted in some way. For example, an audio recording made with poor equipment may be filtered to better represent the sound as it actually occurred.

Both separation and restoration can be achieved using either analog or digital filters so it is useful to compare their features and performance. Analog filters are cheap, fast, and have a large dynamic range in both amplitude and frequency. However, digital filters are vastly superior in terms of filtering performance, i.e. passing desired frequencies and removing signals above/below the cutoff frequency(s). For example, a low-pass digital filter can have a gain of 1 +/- 0.0002 from DC to 1000 hertz and a gain of less than 0.0002 for frequencies above 1001 hertz. The entire transition from pass to cutoff occurs within only 1 hertz. In contrast, opamp hardware filters must spread this cutoff over decades of frequency (~-20dB/decade), allowing significant unwanted signal strength to pass through the filter. This makes a dramatic difference in how filtering problems are approached. With analog filters, the emphasis is on handling limitations of the electronics, such as the accuracy and stability of the resistors and capacitors. In comparison, digital filters are so good that the filtering performance is frequently ignored. Design emphasis can shift to the limitations of the signals and theoretical issues regarding their processing. The cost (compared to hardware filters) is potentially more circuitry needed (microcontrollers are much more complex than opamps), more power, and slower speed.

As shown in Figure 1, every linear filter can be described by an impulse response, a step response and a frequency response. Each of these responses contains complete information about the filter in a different form. If one of the three is specified, the other two can be directly calculated from it. All three of these representations are important, because they describe how the filter will react under different circumstances. The *impulse response* is the output of a system when the input is an impulse. Similarly, the *step response* is the output when the input is a step. The frequency response can be found by taking the discrete Fourier transform (DFT) (e.g., using a fast Fourier transform (FFT) algorithm) of the impulse response. The frequency response can be plotted on a linear vertical axis, such as in Fig. 1b, or on a logarithmic scale (decibels), as shown in Fig. 1d. The linear scale is best at showing the passband ripple and roll-off, while the decibel scale is needed to show the stopband attenuation.

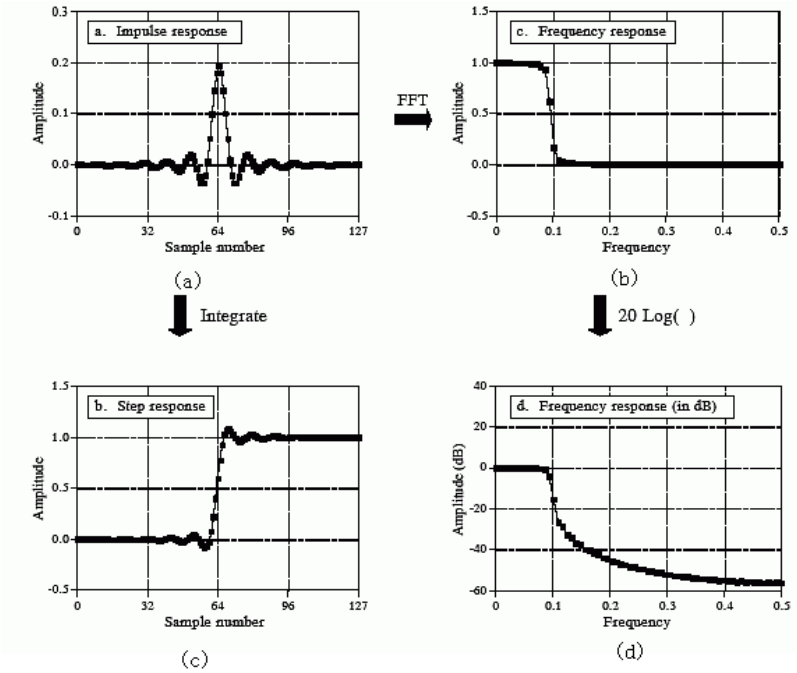


Figure 1. Responses of a linear filter. a) impulse response, b) frequency response, c) step response, and d) frequency response in dB

LABORATORY PROCEDURE:

In this lab, you will first go through tutorial exercises showing how to use LabVIEW and Matlab for filter design. Then you will design your own filters in both software programs for ECG signals.

Part 1: Filter Design in LabVIEW

In this exercise, you are going to implement a digital signal processing filter in LabVIEW.

1. In LabVIEW, search for the "**classical filter design**" VI and put it on the block diagram. In the configuration window, set the following:

Parameter	***Value***		***Parameter***	***Value***
filter type	lowpass		stopband edge frequency	800Hz
sampling frequency	10000Hz		stopband frequency attenuation	60dB
passband edge frequency	100Hz		design method	Equi-Ripple FIR
passband ripple	0.022dB			

After those parameters are set, you should see the frequency response of magnitude and zero-pole distribution on the right. Click OK.

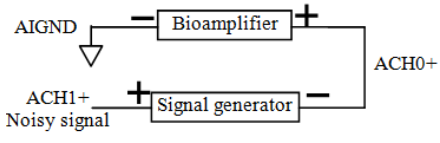
2. Search for the "**filter analysis**" VI and place it on the block diagram. This will more clearly show characteristics of the filter you just implemented. Click OK when the configuration window opens.
3. Wire the "filter out" of the classical filter design to the "filter in" of the filter analysis. Run the program.
4. Double click the filter analysis VI to open the configuration dialog box. You can see the impulse response, step response, magnitude response, phase response, pole-zero plot and group delay of the filter.
5. Search for the "**simulate signal**" VI and place it on the block diagram. In the configuration box, check the "add noise" box, choose "uniform white noise" for the noise type and enter 0.2 for the noise amplitude. Click OK.
6. Search for the "**DFD filtering**" VI and place it on the block diagram. Select single channel->multiple sample->waveform.
7. Connect the output of simulate signal to the signal in of DFD filtering. Connect filter out of classical filter design to filter of DFD filtering.
8. Place a waveform graph on the front panel and connect it to the signal out of the DFD filtering VI in the block diagram. Connect the output of simulate signal VI to the same waveform graph and a merge signals VI will appear at the same time.
9. Place a while loop for the whole block diagram.
10. Run the program. Show the TA both the unfiltered and filtered signal. Ask the TA to sign off on your grading sheet.

Part 2: Filtering noisy ECG signal in LabVIEW

Now you will use the filter tool in LabVIEW to filter the noise from the noisy ECG signal from your body. You need to design your own filter by setting new parameters in the configuration dialog box of the classical filter design VI.

First, we will generate a noisy signal by adding noise (from a signal generator) to the amplified ECG signal from your body.

1. Follow the procedures from Lab 4 to connect the ECG electrodes to one team member's body, and then connect the cables of the isolating amplifier to the electrodes.
2. Connect the output of the isolating amplifier to channel 1 of the ETH-256 Bioamplifier. On the amplifier, set the following parameters: HF = 0.3 , LF = 50, Gain = 1x.
3. On the signal generator, generate a sine wave with 1V Vpp and 500Hz.
4. Connect the signal generator and the output of the amplifier in series as shown in the figure to the right. This will add the noise signal on top of the ECG signal. Connect the negative output of the amplifier to AIGND of ELVIS board. Connect the positive output of the amplifier to ACH0+. Connect the positive output of the signal generator to the ACH1+.



Configuration of noisy ECG signal

Now we will setup LabVIEW to filter the noise out of the noisy ECG signal.

5. Replace the simulate signal VI from Part 1 with the DAQ assistant VI. Add two voltage inputs channels ai0 and ai1. Set the input range max to **5** and input range min **-5**. Set the rate to **2000** and sample to **5000**. Check Lab 3, exercise 1 if you need help setting these parameters.
6. Delete all broken wires. Search for the spilt signals VI. Place it on the block diagram and use the mouse to resize it to two outputs. Connect each output of the split signals with a wave graph.
7. Delete the merge signals VI and all broken wires in the LabVIEW block diagram.
8. Place the write to measurement VI to the block diagram and create a control button for it. Connect the signal 2 output of the split signals VI to the input of the write to measurement VI. Name the filename as "NoisySignal.lvm"
9. Click classical filter design VI and open configuration dialog box. Set the sampling frequency to be **2000**. Determine the rest of the filter parameters needed to remove the noise from the measured ECG signal. Enter these into the configuration window.
10. Run the program. Start to record the noisy ECG signal for a few seconds. Observe the waveform of the filtered signal.
11. When you think you have a good result, write down your filter parameters on the grading sheet and plot the noisy ECG signal in Matlab. You will use this noisy signal later. Show the TA your results and ask him to sign off on your grading sheet.

Part 3: Filter Design in Matlab

Simulink is a program that runs as a companion to MATLAB. Simulink provides a graphical user interface (GUI) that is used in building block diagrams, performing simulations, as well as analyzing results.

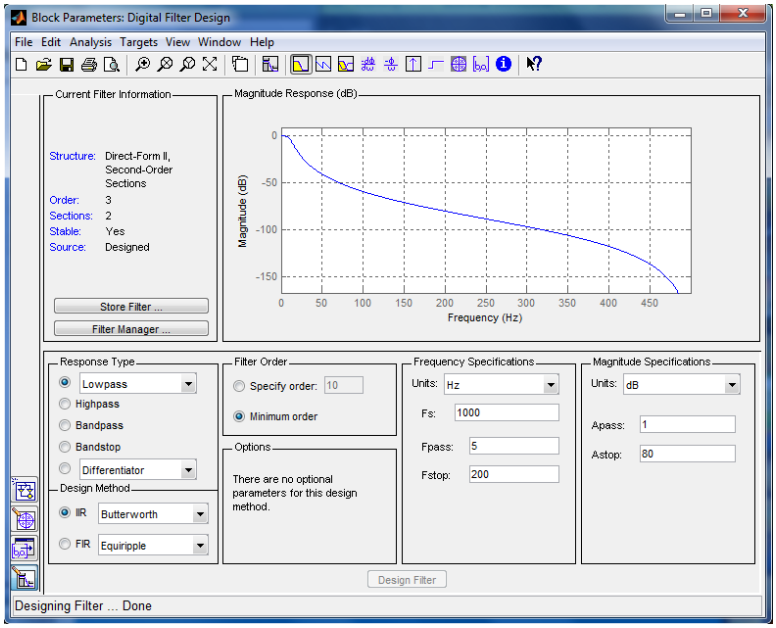
In this section, you will implement a digital signal filter in Matlab/Simulink environment. First we will generate a noisy sine wave and display it.

1. Run Matlab on your computer. In the command window, type "simulink". A simulink Library Browser will pop up.
2. Go to File->New->Model and a new blank model window will appear.
3. In the Simulink Library Browser window, in the left panel, choose Simulink->Sources->Sine Wave and drag it to the model window. Double-click Sine Wave in the model window and a new window for setting parameters will appear. Set the frequency to be **2*pi** and the Sample time to be **0.01**. Click OK.
4. Choose Simulink->Sources->Uniform Random Number and drag it to the model window. Set the Minimum to **-0.1**, Maximum to **0.1** and Sample time to **0.01**. Click OK.
5. Choose Simulink->Math Operations->Add. Connect output of Sine Wave and Uniform Random Number to the inputs of Add block. This will generate a noisy sine wave.
6. Choose Simulink->Sinks->Scope and Connect the output of Add to the Scope. Name it "noisy signal" by clicking its default name "scope".
7. Click the  button in the model window to start the simulation. When finished, double-click on the Scope block and a waveform window will pop up. Maximize the waveform window. Right click on the waveform window and choose Autoscale. Now you should see how noisy it is.

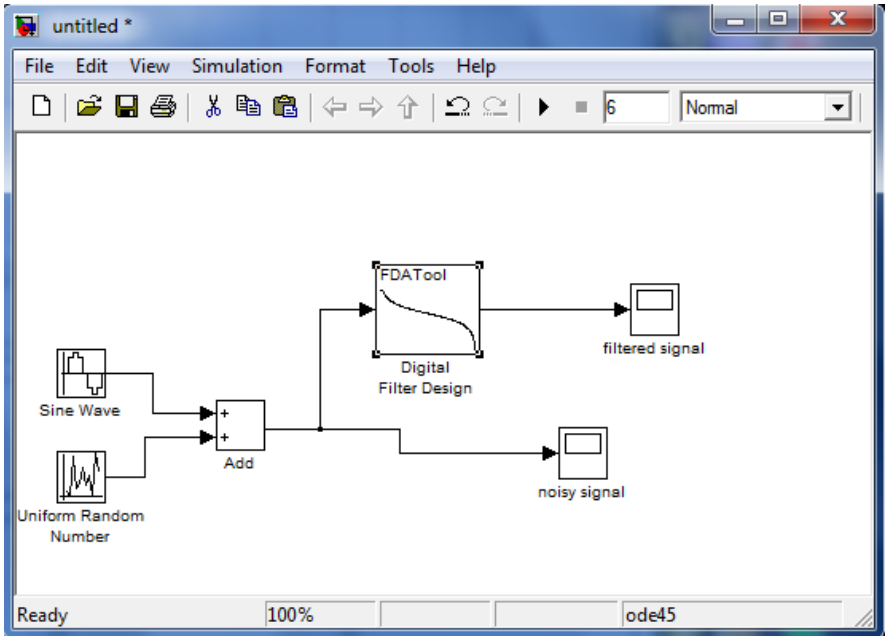
Now we will add a filter to clean up the noise.

8. From Signal Processing Blockset->Filtering->Filter Designs, select Digital Filter Designs. Double-click it to set the parameters. Set Response type to **low pass**, Design Method to **IIR**

butterworth, Units to **Hz**, Fs to **1000**, Fpass to **5** and Fstop to be **200**. The final setup is shown below. Click Design Filter and then close it.



9. Place one more scope on the diagram. Change its name to “filtered signal”. Wire the output of Add to the input of the Digital Design Filter and connect the output of the Digital Design Filter to the second Scope. The final block diagram is shown below.



10. Run the simulation. Show the waveforms of both the noisy signal and the filtered signal.
11. Save the file as part3.mdl.
12. Record a comment on what you observe on the grading sheet. Ask the TA to check off Part 3.

Part 4: ECG Signal Filtering in Matlab

In this part, you will use the noisy ECG signal from part 3 and set filter parameters in Matlab. Results between LabVIEW and Matlab filters will be compared.

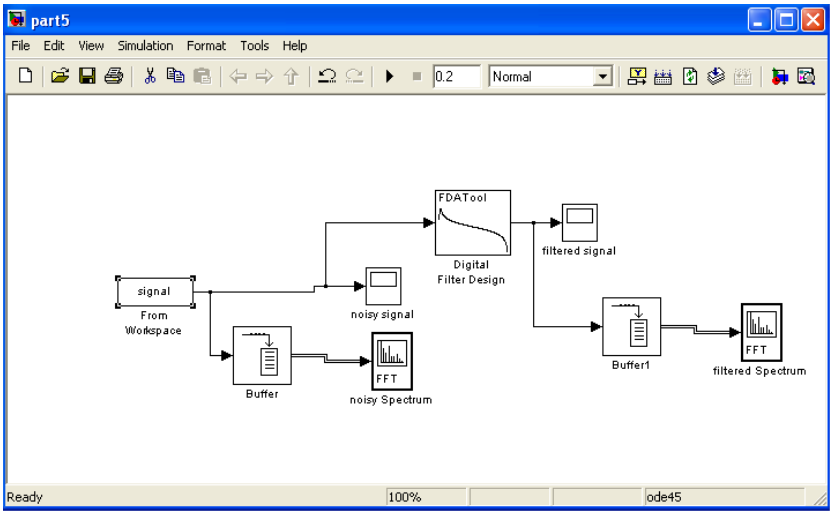
1. In the Matlab command window, type “load NoisySignal.lvm”. Then type “NoisySignal(:,1)=NoisySignal(:,1)-NoisySignal(1,1);” (without the quotes)
2. Open the simulink model “part3.mdl” you just created in part 3 and save it as “part4.mdl”.
3. Choose Simulink->Sources->From Workspace. Double-click it and type “NoisySignal” in Data. In sample time, input **0.0005**.
4. Delete Sine Wave, Uniform Random Number and Add block. Delete all broken wires among them as well. Connect the output of From Workspace to the input of “noisy signal” scope and Digital Filter Design block.
5. Determine what values should be set for filter parameters and then set the parameters of the Digital Filter Design. Set the simulation stop time to be 2.5 and start the simulation. Adjust filter parameters as needed to remove the noise from the desired signal.
6. Compare the result with the result in LabVIEW. Record a comment on the comparison between results on the grading sheet.
7. Show your results to the TA and ask the TA to check off this part.

Part 5: Unknown Signal Filtering in Matlab

Usually the noise and the signal occupy different frequencies and thus look different in the frequency domain. The basic idea of filtering a noisy signal is to transform it into the frequency domain and analyze what frequency range the signal and noise occupy. According to the results of the analysis, the specification of a filter can be designed and implemented to denoise the noisy signal.

In this part, you are given a noisy signal in .mat format and you must analyze this signal by yourself and design a filter to clean up the noisy signal.

1. Download the signal.mat file from the class website. Save it in the current directory of Matlab.
2. Type “load signal.mat” in the command window.
3. Open the simulink model “part4.mdl” and save it as “part5.mdl”.
4. Search Buffer block in the Simulink library browser and place it on the diagram. Click it and type 512 for buffer size.
5. Search for Spectrum Scope and place it on the diagram. Name it as “Noisy spectrum”.
6. Connect the output of From Workspace to the input of Buffer and the output of Buffer to the input of Noisy spectrum.
7. Place another Buffer and Spectrum Scope in the diagram. Click Buffer and type 512 for buffer size. Name the scope as “filtered spectrum”.
8. Connect the output of Digital Filter Design to the input of Buffer and the output of Buffer to the input of filtered spectrum. The final diagram should look like the image below.
9. In the workspace of Matlab, double-click the loaded signal and observe its numbers in the variable editor. Look at the first column, which is time information. Determine what the sampling frequency should be for this signal. Write it down in the grading sheet.



10. Now click From Workspace block, type signal in Data. Set the correct sample time here determined from step 9.
11. Set the simulation stop time to be 0.2. Run the program.
12. Observe the noisy spectrum. Click Axes->Autoscale. What frequency range does the noise occupy? At what frequency range does the useful signal dominate? Write it down on the grading sheet.
13. Now click Digital Filter Design and start designing your own filter to get rid of the noise.
14. Run the program. Observe the filtered signal and the filtered spectrum.
15. When you're satisfied with your result, print configuration dialog box of Digital Filter Design and turn it in with grading sheet.
16. Show the TA your results and ask the TA to check off Part 5.

Wrap Up

1. Once the TA has checked off your circuit, clean up your lab bench.
2. Turn in your Grading Sheet to the TA.